Vulnerability in Xiaomi's Pre-Installed Apps: When Security Is Not What it Seems

Research By: Slava Makkaveev

Smartphones usually come with pre-installed apps, some of which are useful and some that never get used at all. What a user does not expect, however, is for a preinstalled app to be an actual liability to their privacy and security.

Check Point Research recently discovered a vulnerability in one of the preinstalled apps in one of the world's biggest mobile vendors, Xiaomi, which with almost 8% market share rank HYPERLINK "http://gs.statcounter.com/vendor-market-share/mobile/worldwide/2018"s HYPERLINK "http://gs.statcounter.com/vendor-market-share/mobile/worldwide/2018" third in the mobile phone market. Ironically, it was the pre-installed security app, 'Guard Provider' (com.miui.guardprovider), which should protect the phone by detecting malware, which actually exposes the user to an attack.

Briefly put, due to the unsecured nature of the network traffic to and from Guard Provider, a threat actor could connect to the same Wi-Fi network as the victim and carry out a Man-in-the-Middle (MiTM) attack. Then, as part of a third-party SDK update, he could disable malware protections and inject any rogue code he chooses such to steal data, implant ransomware or tracking or install any other kind of malware.

Check Point responsibly disclosed this vulnerability to Xiaomi, which released a patch shortly after.

Fig 1: Xiaomi's preinstalled Security App, known as 'Guard Provider'

Three 3rd Party SDKs in One - How the Attack Works

The Xiaomi 'Guard Provider' is a pre-installed app in all mainstream Xiaomi phones that uses several third-party Software Development Kits (SDKs) as part of the security service it offers, including various types of the device protection, clearing and boosting.
The app includes three different antivirus brands built in that the user can choose from to keep their phone protected: Avast, AVL and Tencent. Upon selecting the app, the user selects one of these providers as the default Anti-Virus engine to scan the device.

Before explaining the potential attack scenario, it is important to point out that there are actually some hidden disadvantages in using several SDKs within the same app. Because they all share the app context and permissions, these main disadvantages are that:

- A problem in one SDK would compromise the protection of all the others.
- The private storage data of one SDK cannot be isolated and can therefore be accessed by another SDK.

With the case of Xiaomi Guard Provider, we show below how a Remote Code Execution (RCE) attack is possible when integrating two SDKs with problematic behavior.

Briefly put, due to Guard Provider's network traffic from any Xiaomi device being unsecured, this allows it to be intercepted via a Man-in-the-Middle (MiTM) attack and inject rogue code as part of a third-party SDK update. Let's look at the possible attack scenario.

Stage 1: Catch the Avast Update

By default, with Avast set as the app's security scanner, the app periodically updates its virus database by downloading the *avast-android-vps-v4-release.apk* APK file to Guard Provider's private directory:
/data/data/com.miui.guardprovider/app_dex/vps_update_<*timestamp*>.apk.

This APK file is then loaded and executed by Avast SDK before the scan of the device and contains the timestamp when the file was downloaded. For example, *vps_update_20190205-124933.apk.*

Figure 2: Avast update file.

Unfortunately, due to the fact that the update mechanism uses an unsecured HTTP connection to download this file, a threat actor, via a MiTM attack, can detect the timing of the Avast update and predict what the disk's APK file name will be next. The attacker needs simply to intercept the response part of the *http://au.ff.avast.sec.miui.com/android/avast-android-vps-v4-release.apk* connection.

Figure 3: Avast update traffic

Hold this thought in mind, as the predicted file name of the Avast update will then be used in the second step of the attack.

By way of the initial interception, the MITM attacker is thus able to prevent future Avast updates by responding with a "404 error" to *http://au.ff.avast.sec.miui.com/android/vps_v4_info.vpx* requests.

Stage 2: Overwrite the Avast Update APK via an AVL Update Path-Traversal Vulnerability

Once the attacker starts actively blocking connections to the Avast server, the user would switch the default antivirus to another one – in this scenario, AVL Anti-Virus. Remember, the AVL antivirus SDK is also a built-in part of the Guard Provider app.

After AVL becomes the default antivirus, it will immediately update the app with its virus database. It does this by checking for the existence of new virus signatures by requesting a configuration file (e.g *http://update.avlyun.sec.miui.com/avl_antiy/miuistd/siglib/20180704.cj.conf*). This .conf file has a plain text format, indicating the URL, size and MD5 hash of the ZIP archive with new signatures.

Figure 4: AVL update config file.

After processing the config file, AVL then downloads the archive of signatures indicated in the *read_update_url* field and decompresses it to the Guard Provider directory.

Once again, the MITM attacker is able to change the content of the .conf file because it was downloaded over a non-secure connection, using *is_new=0* to show the existence of a new update and provide the

URL link to a crafted ZIP file.

Figure 5: Patched AVL update config file.

It turns out that the AVL SDK is vulnerable to another security issue that helps the attacker carry out the second stage of the attack: a path-traversal during the decompression process. As a result, the attacker is able to use a crafted archive to overwrite any file in the app's sandbox, including files related to another SDK.

Therefore, a crafted APK, which is appended as *"../../app_dex/vps_update_20190205-124933.apk"* entry into the ZIP signatures' archive would successfully overwrite the previously downloaded update from Avast, as both anti-virus SDK components use the same sandbox in their respective SDKs.

If you recall, the APK file name of the last Avast update was detected during the first step of the MiTM attack.

Figure 6: Crafted archive with AVL signatures.

All that the attacker needs to do now is release the Avast communication and block AVL's communication until the user again chooses Avast as the active anti-virus. When this happens, the crafted malicious APK file will be loaded and executed by the Avast SDK.

The attack is successful because the previous Avast update's signature file was not verified before loading and Guard Provider has already checked it the first time it was downloaded. It thus assumes there is no reason to verify it again. In this way, the crafted malicious file can be downloaded and run as he has essentially sneaked around the guard's back.

Conclusion

It is completely understandable that users would put their trust in smartphone manufacturers' preinstalled apps, especially when those apps claim to protect the phone itself. This vulnerability discovered in Xiaomi's 'Guard Provider', however, raises the worrying question of who is guarding the guardian. And although the guardian should not necessarily need guarding, clearly when it comes to how apps are developed, even those built in by the smartphone vendor, one cannot be too careful.

The above attack scenario also illustrates the dangers of using multiple SDKs in one app. While minor bugs in each individual SDK can be often be a standalone issue, when multiple SDKs are implemented within the same app it is likely that even more critical vulnerabilities will not be far off.

Check Point SandBlast Mobile, if it were installed on the device, would detect and prevent the initial man-in-the-middle attack, thereby eliminating the threat caused by this vulnerability in Xiaomi’s Guard Provider.